Towards Automated Analysis and Visualization of Distributed Software Systems

Martin Beck
Hasso Plattner Institute
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam, Germany
martin.beck@hpi.uni-potsdam.de

Jürgen Döllner
Hasso Plattner Institute
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam, Germany
juergen.doellner@hpi.uni-potsdam.de

ABSTRACT

This paper sketches one approach to facilitate comprehension of distributed software systems. These systems gain more and more importance due to a paradigm shift in software systems and applications, evolving from single-chip solutions to multi-tiered web-based applications. For a single developer, it becomes increasingly difficult to cope with the complexity of such software systems. We propose a novel automated analysis and visualization technique that aims at the interactive exploration of static structures and behavior of distributed software systems. First, it automatically gathers communication data from the instrumented system components. Second, it generates a visual representation using a heuristic layout approach. Finally, it allows developers to interactively refine and explore this structural and behavioral information.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement—*Restructuring, reverse engineering, and reengineering*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems; D.2.5 [**Software Engineering**]: Testing and Debugging—*tracing*

General Terms

Design, Documentation

Keywords

Distributed Systems, Software Visualization, Dynamic Analysis

1. INTRODUCTION

Distributed software systems gain more and more attention due to the rise of service-based software, which enables reliable, configurable and scalable IT solutions. Consequently, software developers are confronted with the necessity to efficiently develop, maintain and thus understand this category of systems. However, developers generally cannot cope with the inherent complexity of such systems, leading to a severe increase in development and maintenance costs as well as project risks. To improve program comprehension and to gain insights into the distributed system's architecture, dedicated software development tools are required. For example, a web-based shop application may be faced with timeouts in one out of a hundred uses, and developers have to investigate the reason for this behavior.

We propose a novel technique for automated analysis and visualization of distributed systems that extracts, analyzes and visualizes the messages exchanged in the system. Based on an existing software tracer [4], it generates detailed execution trace data for each participating component in the distributed system and, thereby, automatically analyzes its communication behavior. The results are visualized by a heuristically generated visual representation of the system components and their communication behavior using a magnet-driven 2D layout.

This technique enhances understanding of distributed systems by integrating structural and dynamic information in one unified view; it concentrates on the message exchange as an appropriate level of abstraction. To support developers to create a mental model of a system, it is generally not sufficient to understand structural aspects or dynamic aspects in isolation. For this reason, an approach is required that combines both information sources. Interactivity aids to adapt the visual representation to the mental model, leading to improved comprehension.

2. RELATED WORK

Salah and Mancoridis first introduced an approach to gather message data in distributed systems [3]. De Pauw et al. have designed a visualization tool [1] that supports understanding of service-oriented architecture applications. Both approaches follow a model-driven concept, while we head towards an information visualization technique. Spritzer and Freitas have developed a magnet metaphor for interactive graph visualization [5], which we have adapted to our scope. Jerding and Stasko have presented the Information Mural [2], which we use to handle massive message trace data.

3. ANALYSIS

To analyze the structure and behavior of a distributed system, each component is instrumented with a message tracer specific to its communication model, e. g., a Java RMI tracer or an SQL statement tracer. Traced messages are analyzed and classified in categories such as method invocations or database transactions. A central trace server collects each message and stores them for later analysis. Furthermore, the implementation of each component itself is instrumented

SOFTVIS'10, October 25–26, 2010, Salt Lake City, Utah, USA.
ACM 978-1-4503-0028-5/10/10.

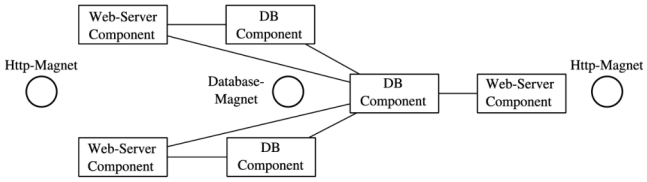

Figure 1: Layout concept using a magnet metaphor. Multiple magnets attract different kinds of nodes, enabling users to adjust the layout interactively.

with a software tracer. The analysis phase recognizes patterns such as queues or worker threads and infers a causal association between incoming and outgoing messages.

Further analysis of the gathered raw data reveals the communication profiles of components and system nodes. Nodes with similar behavior can be clustered and typical design and deployment patterns such as message buses or load balancers are recognized. These are useful hints for the visualization phase. Finally, statistical analysis is used to identify outliers in the message trace data.

4. VISUALIZATION

For visualizing a distributed system's behavior, we compute a dynamic 2D graph layout for the participating components. Existing tools for behavior visualization of distributed systems generally layout components in columns. However, this prevents displaying visual clues about special relationships between components such as spatial proximity of similar software components deployed on multiple nodes.

An initial layout can be automatically derived using the data gathered in the analysis step. It resembles the actual layout of the deployed system. Especially, recognized message patterns help the algorithm to control the layout. To further improve the layout, we investigate a magnet metaphor to visualize the component communication graph. This allows users to interactively adjust the layout to their needs. Fig. 1 illustrates this idea. Multiple magnets attract nodes with regards to their attributes, e. g., nodes sending and receiving HTTP messages or database queries. These magnets can be placed interactively and configured with constraints such as multiple attributes the attracted nodes have to adhere to. Furthermore, the same magnet configuration can be used more than once, thus allowing similar subgraphs to be positioned distinctively.

Distributed systems used in production typically emit millions of messages within a given time frame. Therefore, special care has to be taken to avoid visual clutter. Fig. 2 illustrates our strategy to overcome this. The straight connection line between two nodes separates the available space into two regions, one for each message direction. Incoming and outgoing messages are drawn as bars orthogonal to the separator line in chronological order on their respective side. As the connection line between two components may be rotated in the layout, increasing time is represented by increasing saturation of the message bar colors. Furthermore, incoming and outgoing message bars use different colors to provide a visual distinction regardless of the direction. The height of each bar represents the chosen measurement variable such as message size or transfer time. Due to space constraints, multiple messages can be accumulated in one

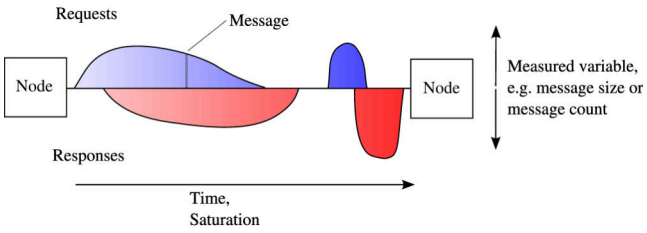

Figure 2: Rendering of traced messages exchanged between two nodes. Increasing saturation represents increasing time, while request and response message directions are differentiated by color. Additional information can be coded in the amplitude.

bar. However, zooming can be used to analyze individual messages. Nevertheless, outliers found during analysis are rendered on top in a special step using another color.

5. CONCLUSIONS

We have presented an approach for program comprehension of distributed software systems. By analyzing message traces, a suitable layout of the participating components is generated automatically. A magnet metaphor allows users of this visualization technique to interactively refine the layout. Categorized message data enhances the structural overview and permits comprehensive exploration of the available data.

Two limitations of the analysis step remain to be addressed in the future: First, the system instrumentation still needs to be deployed manually on each node together with the components to trace. Second, each additional communication and operating system architecture requires a new specific tracer.

A limitation of the message visualization is that it assumes straight connections between communicating nodes. Curved lines, however, would help the layout algorithm to prevent overlapping connections. Another problem represent visible differences in the time scaling due to the different lengths of the connection lines, which complicate the comparison of two communication paths.

6. REFERENCES

[1] W. De Pauw, M. Lei, E. Pring, L. Villard, M. Arnold, and J. F. Morar. Web services navigator: visualizing the execution of web services. *IBM Systems Journal*, 44(4):821–845, 2005.

[2] D. F. Jerding and J. T. Stasko. The information mural: A technique for displaying and navigating large information spaces. *IEEE Transactions on Visualization and Computer Graphics*, 4:257–271, 1998.

[3] M. Salah and S. Mancoridis. Toward an environment for comprehending distributed systems. In *Proceedings of the Working Conference on Reverse Engineering*, pages 238–247. IEEE Computer Society, 2003.

[4] Software Diagnostics Developer Edition. `http://www.softwarediagnostics.com/`, retrieved 14 Jul 2010.

[5] A. S. Spritzer and C. Freitas. A physics-based approach for interactive manipulation of graph visualizations. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 271–278. ACM, 2008.